Why does a ‘numpy.float64’ object return an error stating that it is not iterable in the context of fixing it?

Authored by

stats writer

May 11, 2024

A 'numpy.float64' object is a numerical data type in the numpy library. It is not iterable, meaning it cannot be looped through or broken down into smaller parts. This is why it returns an error when attempting to iterate through it. To fix this error, the object needs to be converted into an iterable data type, such as a list or array, through the use of specific methods. This allows the data to be accessed and manipulated in a more granular way, thus resolving the initial error.

Fix: 'numpy.float64' object is not iterable

One error you may encounter when using NumPy is:

TypeError: 'numpy.float64' object is not iterable

This error occurs when you attempt to perform some iterative operation on a a float value in NumPy, which isn't possible.

The following example shows how to address this error in practice.

How to Reproduce the Error

Suppose we have the following NumPy array:

import numpy as np#define array of data
data = np.array()

#display array of data
print(data)

Now suppose we attempt to print the sum of every value in the array:

```
#attempt to print the sum of every value
for i in data:
print(sum(i))
```

```
TypeError: 'numpy.float64' object is not iterable
```

We received an error because we attempted to perform an iterative operation (taking the sum of values) on each individual float value in the array.

How to Fix the Error

We can avoid this error in two ways:

1. Performing a non-iterative operation on each value in the array.

For example, we could print each value in the array:

```
#print every value in array
for i in data:
print(i)
```

```
1.3
```

1.5

1.6

1.9

2.2

2.5

We don't receive an error because we didn't attempt to perform an iterative operation on each value.

2. Perform an iterative operation on a multi-dimensional array.

```
#create multi-dimensional array
data2 = np.array(, , ])

#print sum of each element in array
for i in data2:
print(sum(i))
```

2.8

3.5

4.7

We don't receive an error because it made sense to use the sum() function on a multi-dimensional array.

In particular, here's how NumPy calculated the sum values:

1.3 + 1.5 = 2.81.6 + 1.9 = 3.52.2 + 2.5 = 4.7

The following tutorials explain how to fix other common errors in Python: